WHAT IS CLAIMED IS:

1. A method for addressing packets, comprising:

receiving, at a first processor, a first packet;

determining as a function of a multidimensional space for representing addresses processed by a set of data processors, a first address for the first packet; and

forwarding the first packet based on the determined first address.

2. The method of claim 1, further comprising:

using an N-tuple space as the multidimensional space.

3. The method of claim 2, further comprising:

assigning to the first processor a first region based on the N-tuple space.

4. The method of claim 3, further comprising:

using the first address, such that the first address represents a point within the first region.

5. The method of claim 4, further comprising:

using N address values as the N-tuple, such that the N address values represent the point.

6. The method of claim 2, further comprising:

using the N-tuple space, such that N is equal to a value of at least two.

7. The method of claim 3, further comprising:

assigning to a second processor a second region based on the N-tuple space, such that the first region is separate from the second region.

8. The method of claim 7, further comprising:

forwarding, at the second processor, a second packet with a second address determined based on the second region, such that the second packet does not conflict with the first packet.

9. The method of claim 7, further comprising:

forwarding, at the second processor, a second packet with a second address determined based on the second region, such that the second address does not conflict with the first address.

10. A method for addressing packets associated with a set of processors, comprising:

receiving, at a first one of the processors, a packet;

reading, at the first processor, an N-tuple address of the received packet;

determining whether the N-tuple address is within an N-tuple space assigned to the first processor;

sending the packet with the N-tuple address, when it is determined that the N-tuple address is within the N-tuple space assigned to the first processor; and

determining a modified N-tuple address, when it is determined that the N-tuple address is not within the N-tuple space assigned to the first processor and sending the packet with the modified N-tuple address.

11. The method of claim 10, wherein the reading step further comprises:

reading as the N-tuple address, a plurality of values from the received packet.

12. The method of claim 11, wherein the reading step further comprises:

reading at least a source port.

13. The method of claim 10, wherein the step of determining whether the N-tuple address is within the N-tuple space, further comprises:

determining whether the N-tuple address is within the N-tuple space based on a comparison between the N-tuple address of the packet and the N-tuple space assigned to the first processor.

14. The method of claim 10, wherein the step of determining whether the N-tuple address is within the N-tuple space, further comprises:

determining whether the N-tuple address of the packet is within the N-tuple space based a quadrant identifier value, wherein the quadrant identifier value corresponds to the first processor.

15. The method of claim 14, wherein the step of determining whether the N-tuple address of the packet is within the N-tuple space, further comprises:

determining the quadrant identifier value based on a hash function.

16. The method of claim 14, wherein the step of determining whether the N-tuple address of the packet is within the N-tuple space, further comprises:

determining the quadrant identifier value based on a hash function and a modulo division.

17. The method of claim 10, wherein the step of determining the modified N-tuple further comprises:

adding a value to the N-tuple address, such that the modified N-tuple address is within the N-tuple space assigned to the first processor.

18. The method of claim 14, wherein the step of determining the modified N-tuple address further comprises:

modifying the N-tuple address based on the quadrant identifier value.

19. The method of claim 10, wherein the step of sending the packet with the N-tuple address, further comprises:

sending the packet with the N-tuple address, such that the packet does not conflict with another N-tuple address associated with a second one of the processors.

20. The method of claim 10, further comprising:

using a firewall as the first processor.

21. The method of claim 10, further comprising:

using a computer as the first processor.

22. The method of claim 10, further comprising:

using a router as the first processor.

23. The method of claim 10, further comprising:

using one or more firewalls as the set of processors, such that the one or more firewalls form a firewall cluster.

24. A method of addressing packets in a firewall cluster, wherein the firewall cluster comprises a set of processors, the method comprising:

receiving, at a first one of the processors, a packet;

reading, at the first processor, an N-tuple address of the received packet;

determining a quadrant identifier based on the read N-tuple address, a hash function, and modulo division;

determining whether the read N-tuple address corresponds to the first processor based on the quadrant identifier;

sending the packet with the N-tuple address, when the quadrant identifier corresponds to the first processor; and

determining a modified N-tuple address, when the quadrant identifier does not corresponds to the first processor and sending the packet with the modified N-tuple address.

25. The method of claim 24, further comprising:

assigning each of the set of processors a firewall node number.

26. The method of claim 25, further comprising:

determining whether the N-tuple address corresponds to the first processor based on the quadrant identifier and the firewall node number.

27. A system for addressing packets, comprising:

means for receiving, at a first processor, a first packet;

means for determining as a function of a multidimensional space for representing addresses processed by a set of data processors, a first address for the first packet; and

means for forwarding the first packet based on the determined first address.

28. A system for addressing packets associated with one or more processors, comprising:

means for receiving, at a first one of the processors, a packet;

means for reading, at the first processor, an N-tuple address of the received packet;

means for determining whether the N-tuple address is within an N-tuple space assigned to the first processor;

means for sending the packet with the N-tuple address, when it is determined that the N-tuple address is within the N-tuple space assigned to the first processor; and

means for determining a modified N-tuple address, when it is determined that the N-tuple address is not within the N-tuple space assigned to the first processor and sending the packet with the modified N-tuple address.

29. A firewall cluster comprising:

means for receiving, at a first one of a set of processors, a packet;

means for reading, at the first processor, an N-tuple address of the received packet;

means for determining a quadrant identifier based on the read N-tuple address, a hash function, and modulo division;

means for determining whether the read N-tuple address corresponds to the first processor based on the quadrant identifier;

means for sending the packet with the N-tuple address, when the quadrant identifier corresponds to the first processor; and

means for determining a modified N-tuple address, when the quadrant identifier does not corresponds to the first processor and sending the packet with the modified N-tuple address.

30. A system, said system comprising:

at least one memory comprising:

code that receives, at a first processor, a first packet;

code that determines as a function of a multidimensional space for representing addresses processed by a set of data processors, a first address for the first packet; and

code that forwards the first packet based on the determined first address; and

at least one processor for executing the code.

31. A system, comprising:

at least one memory comprising

code that receives, at a first one of the processors, a packet;

code that reading, at the first processor, an N-tuple address of the received packet;

code that determines whether the N-tuple address is within an N-tuple space assigned to the first processor;

code that sends the packet with the N-tuple address, when it is determined that the N-tuple address is within the N-tuple space assigned to the first processor; and

code that determines a modified N-tuple address, when it is determined that the N-tuple address is not within the N-tuple space assigned to the first processor and sending the packet with the modified N-tuple address; and

at least one processor for executing the code.

32. The system of claim 31, wherein code that reads further comprises:

code that reads as the N-tuple address, a plurality of values from the received packet.

33. The system of claim 32, wherein code that reads the plurality of values further comprises:

code that reads at least a source port.

34. The system of claim 31, wherein code that determines whether the N-tuple address is within the N-tuple space, further comprises:

code that determines whether the N-tuple address is within the N-tuple space based a comparison between the N-tuple address of the packet and the N-tuple space assigned to the first processor.

35. The system of claim 31, wherein code that determines whether the N-tuple address is within the N-tuple space, further comprises:

code that determines whether the N-tuple address of the packet is within the N-tuple space based a quadrant identifier value, wherein the quadrant identifier corresponds to the first processor.

36. The system of claim 35 wherein code that determines whether the N-tuple address of the packet is within the N-tuple space, further comprises:

code that determines the quadrant identifier value based on a hash function.

37. A firewall cluster comprising:

at least one memory comprising

code that receives, at a first one of a set of processors, a packet;

code that reads, at the first processor, an N-tuple address of the received packet;

code that determines a quadrant identifier based on the read N-tuple address, a hash function, and modulo division;

code that determines whether the read N-tuple address corresponds to the first processor based on the quadrant identifier;

code that sends the packet with the N-tuple address, when the quadrant identifier corresponds to the first processor; and

code that determines a modified N-tuple address, when the quadrant identifier does not corresponds to the first processor and sends the packet with the modified N-tuple address; and

at least one processor for executing the code.

38. A computer program product, the computer program product comprising code for implementing the steps of:

receiving, at a first one of a set of processors, a packet;

reading, at the first processor, an N-tuple address of the received packet;

determining whether the N-tuple address is within an N-tuple space assigned to the first processor;

sending the packet with the N-tuple address, when it is determined that the N-tuple address is within the N-tuple space assigned to the first processor; and determining a modified N-tuple address, when it is determined that the N-tuple address is not within the N-tuple space assigned to the first processor and sending the packet with the modified N-tuple address.

39. The computer program product of claim 38, wherein reading further comprises:

reading as the N-tuple address, a plurality of values from the received packet.

40. The computer program product of claim 39, wherein reading the plurality of values further comprises:

reading at least a source port.

41. The computer program product of claim 39, wherein determining whether the N-tuple address is within the N-tuple space, further comprises:

determining whether the N-tuple address is within the N-tuple space based a comparison between the N-tuple address of the packet and the N-tuple space assigned to the first processor.

42. The computer program product of claim 39, wherein determining whether the N-tuple address is within the N-tuple space, further comprises:

determining whether the N-tuple address of the packet is within the N-tuple space based a quadrant identifier value, wherein the quadrant identifier value corresponds to the first processor.

43. The computer program product of claim 42, wherein determining whether the N-tuple address of the packet is within the N-tuple space, further comprises:

determining the quadrant identifier value based on a hash function.

44. A computer program product, the computer program product comprising code for implementing the steps of:

receiving, at a first one of a set of processors, a packet;

reading, at the first processor, an N-tuple address of the received packet;

determining a quadrant identifier based on the read N-tuple address, a hash function, and modulo division;

determining whether the read N-tuple address corresponds to the first processor based on the quadrant identifier;

sending the packet with the N-tuple address, when the quadrant identifier corresponds to the first processor; and

determining a modified N-tuple address, when the quadrant identifier does not corresponds to the first processor and sending the packet with the modified N-tuple address.

45. A computer program product, the computer program product comprising code for implementing the steps of:

receiving, at a first processor, a first packet;

determining as a function of a multidimensional space for representing addresses processed by a set of data processors, a first address for the first packet; and

forwarding the first packet based on the determined first address.